JUNE
1975
VOL 3
NO 6

27

Popular Computing

B	I	N	G	O
6	22	45	64	81
12	25	49	65	85
13	29	FREE	75	88
16	37	50	76	92
19	38	55	80	94

BINGO

In the game of Bingo, each of many players obtains one or more cards such as the one on the cover. Every card in the game is different. The numbers in the first column (under the B) are more or less randomly chosen in the range from 1 to 20; in the second column in the range 21-40;...in the last column in the range 81-99.

In each game, numbers which include all possibilities are drawn (usually by scrambling 99 plastic balls) in succession. If the number drawn is on a player's card, he marks that square. The first player who gets five squares marked in a row (either horizontally, vertically, or on the two main diagonals) wins. The center square is considered marked on all cards.

A program is to be written to generate the numbers for Bingo cards. A subroutine (such as the one given in PC21) is available to generate random numbers. To illustrate, the numbers in the G column can be created as follows:

1. Generate a random integer.
2. Divide it by 20.
3. Add 61 to the remainder of that division.
4. Repeat steps 1-3 four more times.

So far, the problem of creating Bingo cards is simple and straightforward. Now, several practical constraints are added:

1. On any card, the numbers in any column must not all be consecutive. For example, the set (92, 93, 94, 95, 96) is not allowed in the Ø column.

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $18 per year, or $15 if remittance accompanies the order. For Canada and Mexico, add $4 per year to the above rates. For all other countries, add $6 per year to the above rates. Back issues $2 each. Copyright 1975 by POPULAR COMPUTING.

Publisher: Fred Gruenberger
Editor: Audrey Gruenberger
Associate Editors: David Babcock, Irwin Greenwald
Contributing editors: Richard Andree, Daniel D. McCracken, William C. McGee
Advertising manager: Ken W. Sims
Art Director: John G. Scott
Business Manager: Ben Moore

Reproduction by any means is prohibited by law and is unfair to other subscribers.

@ 2023 This work is licensed under CC BY-NC-SA 4.0

2. In any group of 25 cards produced, all 99 numbers must appear at least once.

3. No two cards should be identical. This constraint needs little attention if the random number generator does its work properly.

PROBLEM 93

Write a program to output any number of patterns for Bingo cards, in sets of 25.

Can Computers fall in love?

Do computers have a sex? Does a computer built under Scorpio get along with a programmer who was born under Capricorn? Could a computer conspiracy ever arise? Could you live a daydream through a computer?

If you've ever thought about these questions before, or if you're first thinking about them now, then it's time you thought about "Creative Computing"—the magazine that speaks your language.

"Creative Computing" is a bi-monthly publication that's about everything that computers are about. From computer poetry to computer art. From the effects of computers on pollution to their effects on privacy. From computers as crime fighters to computers as teaching aids.

"Creative Computing" gives you the chance to be a matador in a bull fight, govern the ancient city of Sumaria, and even fight a space war. Those are only a sample of the kinds of computer games you'll find. Or how about some non-computer games and puzzles?

And that's not all. "Creative Computing" has book reviews, cartoons, fiction, and even a fold-out poster. Plus news and commentary on the twenty computer education projects that have endorsed this publication.

So get involved in the curious world of computers now. Subscribe to "Creative Computing". It's *the* magazine for *the* curious mind.

I'd like to get involved in the curious world of computers. Please enter my subscription to:
Creative Computing, Box 789-M, Morristown, N.J. 07960

☐ 1-Year $8 ☐ 3-Year $21
☐ Payment Enclosed
☐ Please Bill Me (receive one issue less).

☐ Send sample issue $1.00

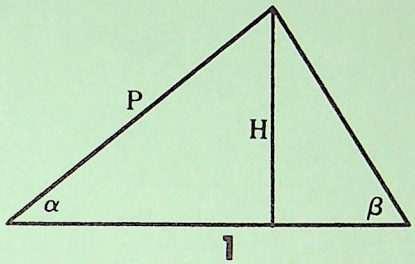
Name_____
Title/Dept _____
School/Company_____
Street Address_____
City_____
State_____ Zip_____

creative computing

The Searchlight problem (Number 89, PC25-11) is reproduced here. The following solution is by Richard Hamming.

The required area is proportional to X^2, so we will set X = 1 for the main solution, and multiply the last result by $X^2/2$.

In a triangle of base 1:



$$\frac{P}{\sin\beta} = \frac{1}{\sin(\pi-\alpha-\beta)}$$

$$P = \frac{\sin\beta}{\sin(\alpha+\beta)}$$

A = Area = H/2

$$2A = H = P\sin\alpha = \frac{\sin\alpha\sin\beta}{\sin(\alpha+\beta)}$$

We write a subroutine Q whose arguments are α and β and whose output is

$$\frac{\sin\alpha\sin\beta}{\sin(\alpha+\beta)}, \qquad \alpha+\beta \neq 0$$

$$0, \qquad \alpha+\beta = 0$$

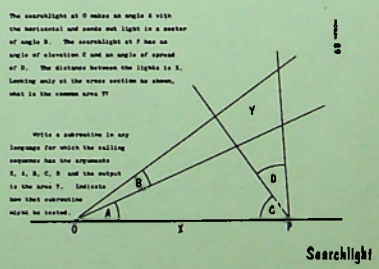
The main subroutine, then, has arguments X, A, B, C, and D with this logic:

1. Check that each of X, A, B, C, and D is greater than or equal to zero, and that A+B+C+D is less than π, else print "no problem."

2. Calculate Y = Q(A+B,C+D) -Q(A+B,C) -Q(A,C+D) +Q(A,C)

3. Return with $X^2/2$ times Y.

Problem Solution



Searchlight

Structured Fortran and COBOL Programs for the Generalized Birthdate Problem

by DANIEL D. McCRACKEN

The generalized birthdate problem was presented in issue No. 25:

> For a population of N things taken at random, how many draws must be made (K) to yield a probability (P) that some two are alike? For N = 365 and P = .5, this is the usual birthdate problem, and K = 23; that is, of 23 people taken at random, it is an even money bet that some two have the same birthdate (day of the year). For N = 100 and P = .8, K = 18; that is, of 18 cars passing at random, the probability is .8 that some two have the last two digits of their license plate alike.

This problem is almost too simple to demonstrate the features and values of structured programming, but it will serve to give some hint of what the subject is about in the limited space available here.

The goal of structured programming is the efficient preparation of correct programs that are easy to understand and easy to work with in testing and maintenance. This, of course, is the goal of any brand of programming. SP (structured programming) attempts to attain this end by two devices, both having to do with program logic, the part of programming that produces the heavily disproportionate fraction of program errors:

1. The programming elements used to express logic are restricted to a small number, like three, which are used in a rigidly consistent way. In the abstract, these are simple sequence, alternation (IFTHENELSE), and iteration; a particular language may express these in different ways, or not have all of them, in which case they must be implemented as best one can. One or two other figures may be added.

2. Consistent indentation is used to make the scope of control of logic statements explicitly clear. Details will vary with the language, of course.

SP also emphasizes matters of good programming practice that have been standard for many years, such as the use of meaningful data names and the organization of the program into rather small segments that communicate with each other only in carefully controlled ways.

The result of the application of these principles is a program in which the structure (that is, the interrelation and interaction of the parts) is as easy to understand as we now know how to make it. A program that is easy to understand is more likely to be correct when first tried, since the original programmer can see what he or she actually did, and because it is easier for other programmers to check. A program that is easy to understand, in part because it is organized into small and largely independent segments, is much easier to modify and maintain; a change in one segment is less likely to cause undesired and unexpected changes (errors) elsewhere.

The actual writing of a structured program, from start of work to completed but uncompiled code, may or may not take less time than with conventional methods. But checkout will average out much shorter, and maintenance is greatly simplified. Considering the rule of thumb that checkout takes a third of the programming time, and considering that in commercial shops maintenance can eat up a majority of the total programming effort, the overall productivity increase can be as high as a factor of five, although usually it is more like double.

SP is not a panacea. It takes a bit of study and practice to become effective at it, although many thousands of ordinary working programmers have discovered that it is not a terribly difficult task. Sometimes the completed program may have to be tuned a bit to take care of efficiency or virtual storage considerations, but experience has shown that these matters are not difficult to handle. A program is not automatically correct or easy to read just because the SP rules have been followed mechanically. Good programmers still write better programs than bad programmers. But everybody's work can be improved by the application of these principles.

SP is easy to teach to beginners. Students coming out of college in future years will know nothing else. The big hurdle, as always, is with the experienced programmers, who tend to pooh-pooh any new development unless the demonstration includes walking on water. They will learn.

Oh yes. GOTOs. In a language which is even half-way suitable for SP, there won't be very many in your programs. SP does not amount to paying bounties for removing GOTOs. It's just that if you use the standard logic figures in an attempt to construct programs that display their structure as clearly as possible, you won't need them, except, of course, to implement logic figures not present in the particular language. Fortran, for instance, does not have a generalized iteration statement, which accordingly requires a GOTO.

Since a structured program is so easy to read, there is generally no need for a flowchart for documentation. But flowcharts of programs of the size and complexity found in real life are impossible to read anyway--even if correct and up to date, which is extremely unlikely. If some program design tool is needed in place of flowcharting, the growing opinion is that some form of pseudocode is the best we have right now. Pseudocode, also called program design language or various other names, has not been standardized and probably never should be. As long as the standard logic figures and indentation are used, with sensible allowance for taking advantage of the features of the programming language that is to be used, the programmer is free to write almost anything he or she wants in the pseudocode, which is one of its advantages. A possible pseudocode for the birthdate problem is shown.

SP is here to stay. Many of the techniques employed are not new at all, with the possible exception of the carefully disciplined way that logic is exhibited, but the emphasis is modern: a process for producing programs that are extremely easy to read, and which (therefore) are much more likely to be correct and very much easier to maintain. The writing of gold star programs will become so common that it will not be newsworthy, and maintenance will cease being a torture device.

You may not see what the shouting is about on a first contact with a small sample. Experienced users say this: write three programs using SP, and you'll be a convert.

PSEUDOCODE

```
Initialize: print headings
DO for all population sizes from 50 to 3000 in steps of 50
     DO for all "odds-factors": 1, 2, 3, 4
          Compute PROBLEVEL = 1/(1 + odds-factor)
          Set PROB to 1
          Set N to 1
          DOUNTIL PROB < PROBLEVEL
               PROB = PROB * (1 - N/population)
               Add 1 to N
          ENDDO
          Move N to answer array
     ENDDO
     Move population to output area
     Write a line
ENDDO
Wrapup operations
```

PC27-8

```
C A PROGRAM FOR THE GENERALIZED BIRTHDATE PROBLEM
C
C VARIABLES
C     POP      THE POPULATION SIZE
C     ODDSF    A FACTOR USED IN COMPUTING THE PROBABILITY LEVEL TO
C              BE ACHIEVED.  ALSO USED AS A SUBSCRIPT.
C     ANS      THE RESULTS, STORED IN AN ARRAY FOR PRINTING
C     PROB     THE PROBABILITY AS IT IS BEING COMPUTED
C     PRBLVL   THE PROBABILITY LEVEL TO BE ACHIEVED
C
      INTEGER*4 POP, ODDSF, ANS(4)
      REAL*4 PROB, PRBLVL
   10 FORMAT ('1', 'POPULATION    .500    .667    .750    .800'//)
   20 FORMAT (' ', I7, 3X, 4I7)
C
      WRITE (6, 10)
      DO 50 POP = 50, 3000, 50
          DO 40 ODDSF = 1, 4
              PRBLVL = 1.0 / (1.0 + FLOAT(ODDSF))
              PROB = 1.0
              N = 1
   30             PROB = PROB * (1.0 - FLOAT(N) / FLOAT(POP))
                  N = N + 1
                  IF ( PROB .GT. PRBLVL ) GO TO 30
              ANS(ODDSF) = N
   40     CONTINUE
          WRITE (6, 20) POP, ANS(1), ANS(2), ANS(3), ANS(4)
   50 CONTINUE
      STOP
      END
```

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.
           PROBABILITIES.
       AUTHOR.
           D D MCCRACKEN.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           C01 IS TO-TOP-OF-PAGE.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT OUTPUT-FILE ASSIGN TO UT-S-PRINTER.

       DATA DIVISION.
       FILE SECTION.
       FD  OUTPUT-FILE
           LABEL RECORDS ARE OMITTED.
       01  OUTPUT-RECORD.
           05  CARRIAGE-CONTROL     PICTURE X.
           05  EDITED-POP           PICTURE BBBZZ99BBB.
           05  ANSWER               PICTURE BBBZZZ9 OCCURS 4 TIMES.
```

```
WORKING-STORAGE SECTION.
01  SUBSCRIPT-ITEM          COMPUTATIONAL SYNCHRONIZED.
    05  SUBS                PICTURE S9999.
01  FLOATING-POINT-ITEMS    COMPUTATIONAL-1.
    05  FLSUBS.
    05  POP.
    05  N.
    05  PROBLEVEL.
    05  PROB.

01  HEADING-LINE.
    05  CARRIAGE-CONTROL    PICTURE X VALUE SPACE.
    05  FILLER              PICTURE X(38)
        VALUE 'POPULATION   .500    .667    .750    .800'.


PROCEDURE DIVISION.
MAIN-LINE-ROUTINE.
    OPEN OUTPUT OUTPUT-FILE.
    WRITE OUTPUT-RECORD FROM HEADING-LINE
        AFTER ADVANCING TO-TOP-OF-PAGE.
    MOVE SPACES TO OUTPUT-RECORD.
    WRITE OUTPUT-RECORD AFTER ADVANCING 2 LINES.
    PERFORM COMPUTE-AND-PRINT-ONE-LINE
        VARYING POP FROM 50 BY 50
            UNTIL POP > 3000.
    CLOSE OUTPUT-FILE.
    STOP RUN.

COMPUTE-AND-PRINT-ONE-LINE.
    PERFORM COMPUTE-ONE-ANSWER
        VARYING SUBS FROM 1 BY 1
            UNTIL SUBS > 4.
    MOVE POP TO EDITED-POP.
    WRITE OUTPUT-RECORD AFTER ADVANCING 1 LINES.

COMPUTE-ONE-ANSWER.
    MOVE SUBS TO FLSUBS.
    COMPUTE PROBLEVEL = 1 / (FLSUBS + 1).
    MOVE 1 TO PROB.
    PERFORM PROBABILITY-COMPUTE
        VARYING N FROM 1 BY 1
            UNTIL PROB < PROBLEVEL.
    MOVE N TO ANSWER (SUBS).

PROBABILITY-COMPUTE.
    COMPUTE PROB = PROB * (1 - N / POP).

****************** END OF PROGRAM *********************
```

Results for the Generalized Birthdate Problem
(See also the earlier article in Issue 25)

POPULATION	.500	.667	.750	.800
50	9	11	12	13
100	13	15	17	18
150	15	19	21	22
200	17	22	24	26
250	19	24	27	29
300	21	26	29	32
350	23	28	32	34
400	24	30	34	36
450	26	32	36	39
500	27	34	38	41
550	28	35	40	43
600	30	37	41	44
650	31	38	43	46
700	32	40	45	48
750	33	41	46	50
800	34	43	48	51
850	35	44	49	53
900	36	45	50	54
950	37	46	52	56
1000	38	48	53	57
1050	39	49	54	59
1100	40	50	56	60
1150	41	51	57	61
1200	42	52	58	63
1250	42	53	59	64
1300	43	54	61	65
1350	44	55	62	66
1400	45	56	63	68
1450	46	57	64	69
1500	46	58	65	70
1550	47	59	66	71
1600	48	60	67	72
1650	49	61	68	73
1700	49	62	69	74
1750	50	63	70	76
1800	51	64	71	77
1850	51	64	72	78
1900	52	65	73	79
1950	53	66	74	80
2000	53	67	75	81
2050	54	68	76	82
2100	55	69	77	83
2150	55	69	78	84
2200	56	70	79	85
2250	57	71	80	86
2300	57	72	80	87
2350	58	72	81	87
2400	58	73	82	88
2450	59	74	83	89
2500	60	75	84	90
2550	60	75	85	91
2600	61	76	85	92
2650	61	77	86	93
2700	62	78	87	94
2750	63	78	88	95
2800	63	79	89	95
2850	64	80	89	96
2900	64	80	90	97
2950	65	81	91	98
3000	65	82	92	99

Constants

LITTLE KNOWN SELDOM USED

CONSTANT	VALUE	UNITS
Speed of light	1.8015446 E 12	Furlongs per fortnight
Length of year	3.1556926 E 13	Microseconds
Odds of getting a straight flush	1.3851695 E-05	----
TV carrier wavelength	1.3106578 E 00	yards
One light-year	1.0340083 E 16	yards
332nd power of 2	8.7490029 E 99	----
U.S. national debt	6.0000000 E 11	dollars
U.S. national debt plus $10,000	6.0000000 E 11	dollars
Daily interest on U.S. national debt	1.3150684 E 08	dollars
Bushel and a peck	7.5706640 E 04	grams
Cubic mile	3.7295829 E 16	cubic centimeters
One second	3.1688765 E-08	years
One microsecond	3.1688765 E-16	centuries
Bicentennial	5.2178571 E 03	fortnights
Barometric pressure	2.2778762 E-02	pounds per sq. mm.
Maximum time between successive Easters	4.0000000 E 02	days

All values are given to 8 significant digits in Fortran E notation.

PC27-11

PC27-12

How To Get Into the Computing Industry

ROBERT REINSTEDT THE RAND CORPORATION

& FRED GRUENBERGER

There are hundreds of introductory and survey courses in computing, catering to those whose interest in the field may be casual, at least at first. A fair percentage of the students in these courses discover that computing as a profession is attractive. They are spurred on, after finding that the subject is intrinsically fascinating and that they have some aptitude for it, by ads and magazine articles that indicate that there is an apparent unending demand for new people and that the salaries are attractive. They then start streaming in to instructors' offices with the question "How do I get into this field?"

It usually turns out that the students are painfully naive about our industry:

> "I've had two whole semesters of computing and I now know everything about how to program and if I just raise my hand, won't the industry flood me with offers?"

--and are also blissfully unaware of how to go about getting any job. They have noticed one thing:

> "All the ads in the Sunday papers call for people with years of experience with specific machine types, specific languages, and specific problem areas, and I don't seem to have any of these. So how can I break into the field?"

Well, let's consider the "no experience" barrier first. Look through the ads for any people in any specific skill, from gas pump jockey on up. They all seem to want experienced people, which should be no great surprise. If you were to advertise for someone to do something, wouldn't you prefer to have experienced people apply? You certainly would--but you might settle for a trainee, especially if the supply is short of your demand. In computing, we frequently have the situation where competent people at all levels are in short supply, and the higher the level, the greater the shortage. Even in rough times, there is always a demand for well qualified people. The computing industry is almost always seeking people with maturity and common sense who understand something about how computers and systems work.

Look at it this way: new computers are being installed at the rate of about 2000 per month, and each of these machines requires, on the average, about 6 people to serve it (counting programmers, keypunch operators, console operators, servicemen, salesmen, etc.). Thus, at the start of a semester course, one can predict a need by the end of the course for some 60,000 new people. The total output of all our schools, public and private, in any period of 5 months, probably does not exceed 10,000 people. There's a gap there, and a lot of companies that advertise "Programmer wanted; must have two years' experience in COBOL on System/360 in banking work" are going to have to settle for less.

The student, having no experience to speak of, usually regards the whole situation as unfair. The solution, of course, is to apply for jobs without regard to the stated constraints, but in a planned fashion. Meanwhile, one might consider the other side of the coin. Suppose, somehow, you do surmount the barrier and land a job without actual experience. Then consider how the situation will look six months or a year later. Those ads in the Sunday papers will take on a fresh look, and the unfairness of the situation will become greatly attenuated. With a whole year's experience under your belt, you might even become one of those who complains about all the wet-behind-the-ears youngsters that they're letting in to the field these days.

All right. The student doesn't really believe any of that, and is still wondering how to bring himself to the attention of those in industry who are supposed to be so eager for his services. At this point, he should compose answers to some questions that will give some direction to his job quest:

1. What part of the industry do you want to get into? Users or vendors? Sales, marketing, service, or programming? What kind of programming? Scientific or commercial? Applications or systems?

2. Do you want to work for a small, medium, or large firm (or do you want to run your own business)? There are advantages and disadvantages both ways, but the real question is whether you prefer the big-frog-small-pond atmosphere, or the "organization man" approach. Do you want to maximize security, or the chance at a high salary? Do you have a picture in your mind of your inherent genius letting you sleep late in the morning? As Herb Grosch points out, the only guys in this business who sleep late in the morning are those who have been up most of the night working like dogs.

3. How mobile are you? Good jobs that lie within 20 minutes' drive of where you now live are likely to be scarce. The joke that says that IBM stands for "I've Been Moved" is funny only up to a point.

4. Just what do you have to offer? You've taken courses, and done the homework, and passed the tests--but how much computing have you done? Make a list of all the programs you've written. Do you know anything at all about machines other than the one you've learned on, or languages other than your training language?

5. How much computing lore that was not included in your regular course work have you absorbed? What books have you read besides your texts? Which of the industry journals (there are at least 30 periodicals devoted entirely to computing) have you been reading regularly? Can you read DATAMATION intelligently?

6. Where do you want to be five years from now? At the moment, the idea of writing real programs to massage real data on a real production basis may be fascinating. Will it fascinate you after five years of it? Do you have any idea of how programmers work from day to day?

Step one in attacking the job market is the preparation of a resume. There are several books on this subject, complete with sample resumes that you can paraphrase.* Having an intelligently made resume will do several things for you:

1. It will force you to write down the facts about yourself and what it is you want.

2. It will demonstrate to prospective employers that you mean business and know how to package a proposal.

3. When there are 30 applicants for one job (and that's the kind of job you want to apply for) the ones who have neat and unusual resumes will be remembered. (Consider the student who produced his resume as output of a Fortran program; the resume was an actual listing from the computer.)

4. A printed resume will save you lots of time when you go to fill out application forms. For all the spaces that call for factual information, you can write "See resume attached."

The line above says "printed resume." You can, of course, type many copies, or make Xerox copies. Consider, though, that straight typing can be offset printed for about $3 per hundred copies, and will then look much neater and more professional. If you can include also a clear photo, the cost will be a little higher, but it's well worth it. You want to be remembered, and a photo is a great help. At this point, you're in the situation where a few dollars of investment may pay off a hundredfold.

*For example, Personal Resume Preparation, by Michael P. Jaquish, Wiley, 1968.

For almost any job at all, you will have to have a personal interview. This may be as short as 15 minutes, or as long as 8 hours. It may be conducted by a professional interviewer, or it may be a casual appearing thing conducted by the DP manager who needs a new programmer. In any event, it will appear to you as a brutal inquisition, highly conducive to extreme nervousness and sweating on your part.

The game you will play in an interview is "How has your company managed so far without me?" The game they will play is "Can we trust this guy with our master files? Do we first have to teach him what a file is? And will he be an honest and hard-working employee?"

Interviews come in three grades: kindly, straightforward, and stress. The more desirable the job, the more the interview will lean toward the stress end. (For a job as a keypunch trainee, the interview will not only be kindly, but will tend toward "Won't you please consider us as a place to work?") For a job as a chief systems programmer, it will be more of the nature of "Tell us what you've done in the last year--and it better be good."

It is a game, and there are rules for it, and you should know them. No, you shouldn't ask about the pension plan. If the subject is to be discussed, let the interviewer bring it up. In fact, let him carry the ball most of the time. When he gets to the point of saying "Now tell me about your goals," then you can babble away. There's a fine line between answering the interviewer's questions clearly and succinctly; appearing sullen and uncommunicative; and sounding garrulous.

The rules change somewhat in a stress interview, where the objective of the interviewer is to put the heat on, to see how you act when things get rough. It can all be summed up in a cliche: "If you can keep your head when all about you are losing theirs, then you are a hire, my son." The interviewer is trying to get you rattled in a stress interview and your objective is to maintain your pride and dignity and not get angry. It can get to the point where you must say "I don't think that that's any of your concern, sir," so you should be prepared to say it.

The authors conducted some mock interviews with members of an advanced computing class at California State University, Northridge. The students were given the following information:

Here is the newspaper ad you are answering

PROGRAMMER WANTED. Experience with 360-series computers desirable, but will consider competent trainee. Salary commensurate with background. Work entails programming in high level language; some systems analysis; general project responsibility for computational work in leading steel company. Send resume. Interview will be arranged. Write Box 1234, The Times. Jones Steel Co.

With a little digging, you could obtain information like the following for almost any company:

The Jones Steel Company has the usual business problems, including payroll, accounts receivable, accounts payable, inventory control, and purchasing. Scientific computation is done on problems of steel fabrication, such as stress analysis, heat transfer, and assembly line balancing.

The company has a 360/40 and a 360/30 in separate installations, but there is some sharing of facilities. They have 2500 hourly paid employees (paid each week) and 300 salaried employees (paid monthly). The company has experienced a steady growth rate, in both income and personnel, of 10% per year for over a decade.

The company's offices are in Commerce, about 5 miles east of the freeway interchange. Normal working hours are 8 to 4:30 with a half hour for lunch. All hiring is done by their personnel department.

Three volunteers agreed to be interviewed for this job, and tagged themselves for the kindly, straightforward, or stress interviews. The interviews were videotaped and later played back (with many interruptions for post mortem discussion). The experience seemed to be an eye-opener for everyone concerned, including us.

The value to the student should be obvious. Rehearsal of any drama is vital to giving a good performance on opening night, and in job interviewing every minute is opening night you you're the star. The script can't be memorized, but the more practice, the better. The practice can be nerve-racking and painful (especially if you can see yourself through the cold eye of a TV camera) but is priceless.

There are many companies specializing in recruiting personnel in data processing. They seek resumes from job seekers, and attempt to then match those people with requests from companies. The service is generally free to the applicant and is usually presented to imply that there are thousands of jobs on their lists, thus greatly widening the range of possibilities that may be available to you. The following points should be considered:

1. If the agency is large and well established, the claim for enlarged scope in job-seeking is probably valid. If the match between jobs and people is done by computer (several firms offer this service, or claim to) there is the additional advantage that every appropriate applicant will be considered objectively for every job.

2. Someone must pay for the service. The agencies charge the employer for each job filled, at rates that may reach 15% of the first year's salary. Thus, if you're going for a $10,000 job, the employer must weigh your prospects (which will now include a $1500 charge) against those of other applicants.

3. Offsetting this is the fact that the agency has presumably done some preliminary screening on job-seekers, and this serice is valuable to the employer.

4. Whether or not to use an employment agency is thus one of the decisions that you must make. At the least, our advice is to use these services judiciously; do not shotgun your attack on the market.

Of course, we've assumed all along that if you want to get into the computing industry you're prepared to offer something to the industry; that you're now in some sense a programmer, or you're a good technician, or you think you can sell, or manage, or whatever. Then the idea we're trying to get across is PLAN AHEAD. We see too many students who approach the business of job hunting with about the same amount of preparation they would devote to a blood test. The decisions they are about to make (or have made for them) will not only affect their lives but can make a difference amounting to tens of thousands of dollars. It would seem worth a few hours of intelligent preparation, particularly in a field that puts some emphasis on intelligence. □

N-SERIES 27

Log 27	1.431363764158987311885083709765345927600386592572088
Ln 27	3.295836866004329074185735710767577113942471673468248
$\sqrt{27}$	5.196152422706631880582339024517617100828415761431144
$\sqrt[5]{27}$	1.933182044931762751524878943266168146061860628402307
$\sqrt[10]{27}$	1.390389170315909340485254294616067750994397440802535
$\sqrt[100]{27}$	1.033507512043090049195797748245168264157558105353326
e^{27}	532048240601.798616683747304341177441659255804283688808837731150535969409966953197164611553833
π^{27}	26487841119103.630227110041429076367875996012662446298188316561477908765007259980337211210 8
$\tan^{-1}$ 27	1.533776210920966592236806242071952261342364339364293

Problem 92 (PC26-16) was given incorrectly. For the stated equation ($J^5 + K^5 + L^5 + M^5 = N^5$), Lander and Parkin found one solution in integers, with the value of N less than 150. Four solutions were found for the equation

$$I^5 + J^5 + K^5 + L^5 + M^5 = N^5$$

with the value of N, in one case, less than 75. Our apologies to Lander and Parkin, and to Prof. Richard Andree, who submitted the problem. □

Printed by Argold Press Inc.
16921 Ventura Boulevard, Encino

? FRUSTRATED

. . . SEARCHING FOR INFORMATION?
TIME AND MONEY AT A PREMIUM?
YOU NEED **DATA PROCESSING DIGEST**

Continuous search through computer, management and trade magazines, reports, information services. Readable digests of the best material we find. Book reviews, monthly and yearly index, calendar, complete reference to sources. 12 monthly issues, $51.

Data Processing Digest
Published Each Month Since 1955

Write for Information

Name _____
Dept. _____
Company _____
Address _____
City _____ State _____ Zip _____